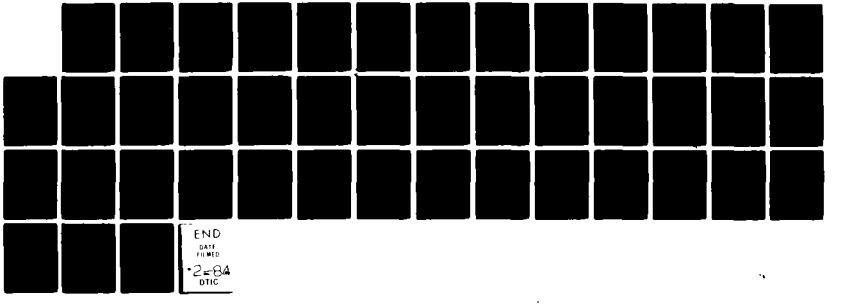
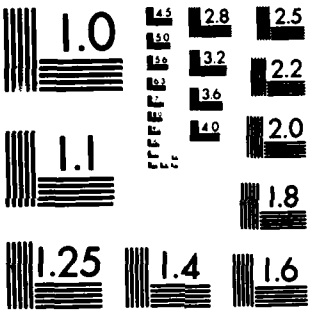
AD-A135 875 CONVERGENCE THEORIES OF DISTRIBUTED ITERATIVE PROCESS: 1/1
A SURVEY(U) MASSACHUSETTS INST OF TECH CAMBRIDGE LAB
FOR INFORMATION AND D.. D P BERTSEKAS ET AL. DEC 83
UNCLASSIFIED LIDS-P-1342 N00014-75-C-1183 F/G 12/1 NL



END
DATE FILMED
2-84
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12

LIDS-P-1342

December, 1983

AD-A135 875

CONVERGENCE THEORIES OF DISTRIBUTED ITERATIVE PROCESS: A SURVEY

by

Dimitri P. Bertsekas*
John N. Tsitsiklis**
Michael Athans*

DTIC
ELECTE
DEC 15 1983
S D
D

ABSTRACT

The authors consider a model of distributed iterative algorithms whereby several processors participate in the computation while collecting, possibly stochastic information from the environment or other processors via communication links. Several applications in distributed optimization, parameter estimation, and communication networks are described. Issues of asymptotic convergence and agreement are explored under very weak assumptions on the ordering of computations and the timing of information reception. Progress towards constructing a broadly applicable theory is surveyed.

DTIC FILE COPY

---

* The research of D.P. Bertsekas was supported by NSF-ECS-8217668 and under DARPA Grant ONR-N00014-75-C-1183. The research of J.N. Tsitsiklis and M. Athans was supported by ONR-N00014-77-C-0532(NR- 041-519).

* Dept. of Electrical Engineering and Computer Science, Laboratory for Information and Decision Systems, M.I.T., Cambridge, Mass. 02139.

** Dept. of Electrical Engineering, Stanford University, Stanford, California.

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

83 12 13 272

1. Introduction

Classical (centralized) theories of decision making and computation deal with the situation in which a single decision maker (man or machine) possesses (or collects) all available information related to a certain system and has to perform some computations and/or make a decision so as to achieve a certain objective. In mathematical terms, the decision problem is usually expressed as a problem of choosing a decision function that transforms elements of the information space into elements of the decision space so as to minimize a cost function. From the point of view of the theory of computation, we are faced with the problem of designing a serial algorithm which actually computes the desired decision.

Many real world systems however, such as power systems, communication networks, large manufacturing systems, public or business organizations, are too large for the classical model of decision making to be applicable. There may be a multitude of decision makers (or processors), none of which possesses all relevant knowledge because this is impractical, inconvenient, or expensive due to limitations of the system's communication channels, memory, or computation and information processing capabilities.

In other cases the designer may deliberately introduce multiple processors into a system in view of the potential significant advantages offered by distributed computation. For problems where processing speed is a major bottleneck distributed computing systems may offer increases in throughput that are either unattainable or prohibitively expensive using a single processor. For problems where reliability or survivability is a major concern distributed systems can offer increased fault tolerance or more graceful performance degradation in the face of various kinds of

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By Per DTIC Form 50
Distribution/on file
Availability Codes
Avail and/or
Dist Special
A/1
DTIC
COPY
INSPECTED

equipment failures. Finally as the cost of computation has decreased dramatically relative to the cost of communication it is now advantageous to trade off increased computation for reduced communication. Thus in database or sensor systems involving geographically separated data collection points it may be advantageous to process data locally at the point of collection and send condensed summaries to other points as needed rather than communicate the raw data to a single processing center.

For these reasons, we will be interested in schemes for distributed decision making and computation in which a set of processors (or decision makers) eventually compute a desired solution through a process of information exchange. It is possible to formulate mathematically a distributed decision problem whereby one tries to choose an "optimal" distributed scheme, subject to certain limitations. For example, we may impose constraints on the amount of information that may be transferred and look for a scheme which results in the best achievable decisions, given these constraints. Such problems have been formulated and studied in the decentralized control context [21,22], as well as in the computer science literature [23,24]. However, in practice these turn out to be very difficult, usually intractable problems [25,26]. We, therefore, choose to focus on distributed algorithms with a prespecified structure (rather than try to find an optimal structure): we assume that each processor chooses an initial decision and iteratively improves this decision as more information is obtained from the environment or other processors. By this we mean that the ith processor updates from time to time his decision x^i using some formula

$$x^i \leftarrow f^i(x^i, I^i) \tag{1.1}$$

where I^i is the information available to the ith processor at the time of the update. In general there are serious limitations to this approach the most obvious

of which is that the function f^i in (1.1) has to be chosen a priori on the basis of ad hoc considerations. However there are situations where the choice of reasonable functions f^i is not too dificult, and iterations such as (1.1) can provide a practical approach to an otherwise very difficult problem. After all, centralized counterparts of processes such as (1.1) are of basic importance in the study of stability of dynamic systems, and deterministic and stochastic optimization algorithms.

In most of the cases we consider the information I^i of processor i contains some past decisions of other processors. However, we allow the possibility that some processors perform computations (using (1.1)) more often than they exchange information, in which case the information I^i may be outdated. This allows us to model situations frequently encountered in large systems where it is difficult to maintain synchronization between various parts of the decision making and information gathering processes.

There are a number of characteristics and issues relating to the distributed iterative process (1.1) that either do not arise in connection with its centralized counterpart or else appear in milder form. First there is a graph structure characterizing the interprocessor flow of information. Second there is an expanded notion of the state of computation characterized by the current results of computation x^i and the latest information I^i available at the entire collection of processors i. Finally when (as we assume in this paper) there is no strict sequence according to which computation and communication takes place at the various processors the state of computation tends to evolve according to a point-to-set mapping and possibly in a probabilistic manner since each state of computation may give rise to many other states depending on which of the processors executes iteration (1.1) next and depending on possibly random exogenous information made available at the processors during execution of the algorithm.

From the point of view of applications, we can see several possible (broadly defined) areas. We discuss below some of them, although this is not meant to be an exhaustive list.

a) Parallel computing systems, possibly designed for a special purpose, e.g. for solving large scale mathematical programming problems with a particular structure. An important distinguishing feature of such systems is that the machine architecture is usually under the control of the designer. As mentioned above, we will assume a prespecified structure, thereby bypassing issues of architectural choice. However, the work surveyed in this paper can be useful for assessing the effects of communication delays and of the lack of synchronization in some parallel computing systems. Some of the early work on the subject [10],[11] is motivated by such systems. For a discussion of related issues see [7].

b) Data Communication Networks. Real time data network operation lends itself naturally to application of distributed algorithms. The structure needed for distributed computation (geographically distributed processors connected by communication links) is an inherent part of the system. Information such as link message flows, origin to destination data rates, and link and node failures is collected at geographically distributed points in the network. It is generally difficult to

implement centralized algorithms whereby a single node would collect all information needed, make decisions, and transmit decisions back to the points of interest. The amount of data processing required of the central node may be too large. In addition the links over which information is transmitted to and from the central node are subject to failure thereby compounding the difficulties. For these reasons in many networks (e.g. the ARPANET) algorithms such as routing, flow control, and failure recovery are carried out in distributed fashion [1]-[5]. Since maintaining synchronization in a large data network generally poses implementation difficulties these algorithms are often operated asynchronously.

c) Distributed Sensor Networks and Signal Processing. Suppose that a set of sensors obtain noisy measurements (or a sequence of measurements) of a stochastic signal and then exchange messages with the purpose of computing a final estimate or identifying some unknown parameters. We are then interested in a scheme by which satisfactory estimates are produced without requiring that each sensor communicates his detailed information to a central processor. Some approaches that have been tried in this context may be found in [27,28,29,30].

d) Large Decentralized Systems and Organizations. There has been much interest, particularly in economics, in situations in which a set of rational decision makers make decisions and then update them on the basis of new information. Arrow and Hurwicz [31] have suggested a parallelism between the operation of an economic market and distributed computation. In this context the study of distributed algorithms may be viewed as an effort to model collective behavior. Similar models have been proposed for biological systems, [32]. Alternatively, finding good distributed algorithms and studying their communication requirements may yield insights on good ways of

designing large organizations. It should be pointed out that there is an open debate concerning the degree of rationality that may be assumed for human decision makers. Given the cognitive limitations of humans, it is fair to say that only relatively simple algorithms can be meaningful in such contexts. The algorithms considered in this paper tend to be simple particularly when compared with other algorithms where decision makers attempt to process optimally the available information.

There are several broad methodological issues associated with iterative distributed algorithms such as correctness, computation or communication efficiency, and robustness. In this paper we will focus on two issues that generally relate to the question of validity of an algorithm.

a) Under what conditions is it possible to guarantee asymptotic convergence of the iterates x^i for all processors i, and asymptotic agreement between different processors i and j $[(x^i - x^j) \to 0]$?

b) How much synchronization between processor computations is needed in order to guarantee asymptotic convergence or agreement?

Significant progress has been made recently towards understanding these issues and the main purpose of this paper is to survey this work. On the other hand little is known at present regarding issues such as speed of convergence, and assessment of the value of communicated information in a distributed context. As a result we will not touch upon these topics in the present paper. Moreover, there are certain settings (e.g., decentralized control of dynamical systems, dynamic routing in data networks) in which issues of asymptotic convergence and agreement do not arise. Consequently, the work surveyed here is not of direct relevance to such situations.

In the next two sections we formulate a model of distributed asynchronous iterative computation, and illustrate its relevance by means of a variety of examples

from optimization, parameter estimation, and communication networks. The model bears similarity to models of chaotic relaxation and distributed asynchronous fixed point computation [10]-[13] but is more general in two respects. First we allow two or more processors to update separately estimates of the same coordinate of the decision vector and combine their individual estimates by taking convex combinations, or otherwise. Second we allow processors to receive possibly stochastic measurements from the environment which may depend in nonlinear fashion on estimates of other processors. These generalizations broaden a great deal the range of applicability of the model over earlier formulations.

In Sections 4 and 5 we discuss two distinct approaches for analyzing algorithmic convergence. The first approach is essentially a generalization of the Lyapounov function method for proving convergence of centralized iterative processes. The second approach is based on the idea that if the processors communicate fast relative to the speed of convergence of computation then their solution estimates will be close to the path of a certain centralized process. By analyzing the convergence of this latter process one can draw inferences about the convergence of the distributed process. In Section 5 we present results related primarily to deterministic and stochastic descent optimization algorithms. An analysis that parallels Ljung's ODE approach [37],[38] to recursive stochastic algorithms may be found in [35] and in a forthcoming publication. In Section 6 we discuss convergence and agreement results for a special class of distributed processes in which the update of each processor, at any given time, is the optimal estimate of a solution given his information, in the sense that it minimizes the conditional expectation of a common cost function.

2. A Distributed Iterative Computation Model

In our model we are given a set of feasible decisions X and we are interested in finding an element of a special subset X^* called the solution set. We do not specify X^* further for the time being. An element of X^* will be referred to as a solution. Without loss of generality we index all events of interest (message transmissions and receptions, obtaining measurements, performing computations) by an integer time variable t. There is a finite collection of processors i=1,...,n each of which maintains an estimate $x^i(t) \in X$ of a solution and updates it once in a while according to a scheme to be described shortly. The ith processor receives also from time to time m_i different types of measurements and maintains the latest values $z_1^i, z_2^i, \ldots, z_{m_i}^i$ of these measurements. (That is, if no measurement of type j is received at time t, then $z_j^i(t+1) = z_j^i(t)$). The measurement z_j^i is an element of a set Z_j^i. Each time a measurement z_j^i of type j is received by processor i the old value z_j^i is replaced by the new value and the estimate x^i is updated according to

$$x^i(t+1) = M_{ij}(x^i(t), z_1^i(t), \ldots, z_{m_i}^i(t)) , \tag{2.1}$$

where M_{ij} is a given function. Each node i also updates from time to time the estimate x^i according to

$$x^i(t+1) = C_i(x^i(t), z_1^i(t), \ldots, z_{m_i}^i(t)) \tag{2.2}$$

where C_i is a given function. Thus at each time t each processor i either receives a new measurement of type j and updates x^i according to (2.1), or

updates x^i according to (2.2), or remains idle in which case $x^i(t+1) = x^i(t)$ and $z_j^i(t+1) = z_j^i(t)$ for all j. The sequence according to which a processor executes (2.1) or (2.2) or remains idle is left unspecified and indeed much of the analysis in this paper is oriented towards the case where there is considerable a priori uncertainty regarding this sequence. One of the advantages of this approach is that difficult analytical problems arising due to consideration of non-classical information patterns [21] do not appear in our framework. Note that neither mapping M_{ij} or C_i involves a dependence on the time argument t. This is appropriate since it would be too restrictive to assume that all processors have access to a global clock that records the current time index t. On the other hand the mappings M_{ij} and C_i may include dependences on local clocks (or counters) that record the number of times iterations (2.1) or (2.2) are executed at processor i. The value of the local counter of processor i may be artificially lumped as an additional component into the estimate x^i and incremented each time (2.1) or (2.2) are executed.

Note that there is redundancy in introducing the update formula (2.2) in addition to (2.1). We could view (2.2) as a special case of (2.1) corresponding to an update in response to a "self-generated" measurement at node i. Indeed such a formulation may be appropriate in some problems. On the other hand there is often some conceptual value in separating the types of updates at a processor in updates that incorporate new exogenous information (cf. (2.1)), and updates that utilize the existing information to improve the processor's estimate (cf. (2.2)).

The measurement $z_j^i(t)$, received by processor i at time t, is related to the processor estimates $x^1, x^2, \ldots, x^n$ according to an equation of the form

$$z_j^i(t) = \phi_{ij}(x^1(\tau_j^{i1}(t)), x^2(\tau_j^{i2}(t)), \ldots, x^n(\tau_j^{in}(t)), \omega), \qquad (2.3)$$

where ω belongs to the sample space Ω corresponding to a probability space (Ω, F, P).

We allow the presence of delays in equation (2.3) in the sense that the estimates $x^1,\ldots,x^n$ may be the ones generated via (2.1) or (2.2) at the corresponding processors at some times $\tau_j^{ik}(t) \leq t$, prior to the time t that $z_j^i(t)$ was received at processor i. Furthermore the delays may be different for different processors. We place the following restriction on these delays which essentially says that successive measurements of the same type depend on successive processor estimates.

Assumption 2.1: If $t \geq t'$, then

$$\tau_j^{ik}(t) \geq \tau_j^{ik}(t'), \quad \forall i,j,k \quad .$$

For the time being, the only other assumption regarding the timing and sequencing of measurement reception and estimate generation is the following:

Assumption 2.2 (Continuing Update Assumption): For any i and j and any time t there exists a time $t'>t$ at which a measurement z_j^i of the form (2.3) will be received at i and the estimate x^j will be updated according to (2.1). Also for any i and time t there exists a time $t''>t$ at which the estimate x^i will be updated according to (2.2).

The assumption essentially states that each processor will continue to receive measurements in the future and update his estimate according to (2.1) and (2.2). Given that we are interested in asymptotic results there isn't much we can hope to prove without an assumption of this type. In order to formulate substantive convergence results we will also need further assumptions on the nature of the mappings M_{ij}, C_{ij}, and ϕ_{ij} and possibly on the relative timing of measurement

receptions, estimate updates and delays in (2.3) and these will be introduced later. In the next section we illustrate the model and its potential uses by means of examples.

It should be pointed out here that the above model is very broad and may capture a large variety of different situations, provided that the measurements z_j^i are given appropriate interpretations. For example, the choice $z_j^i(t) = x^j(\tau_j^{ij}(t))$ corresponds to a situation where processor i receives a message with the estimate computed by processor j at time $\tau_j^{ij}(t)$, and $t-\tau_j^{ij}(t)$ may be viewed as a communication delay. In this case processors act also as sensors generating measurements for other processors. In other situations however specialized sensors may generate (possibly noisy and delayed) feedback to the processors regarding estimates of other processors of (cf. (2.3)). Examples of both of these situations will be given in the next section.

3. Examples

An important special case of the model of the previous section is when the feasible set X is the Cartesian product of n sets

$$X = X_1 \times X_2 \times \ldots \times X_n ,$$

each processor i is assigned the responsibility of updating the ith component of the decision vector $x = (x_1, x_2, \ldots, x_n)$ via (2.2) while receiving from each processor j ($j \neq i$) the value of the jth component x_j. We refer to such distributed processes as being specialized. The first five examples are of this type.

Example 1: (Shortest Path Computation)

Let (N,A) be a directed graph with set of nodes $N=\{1,2,\ldots,n\}$ and set of links A. Let $N(j)$ denote the set of downstream neighbors of node i, i.e. the nodes j such that (i,j) is a link. Assume that each link (i,j) is assigned a positive scalar a_{ij} referred to as its length. Assume also that there is a directed path to node 1 from every other node. Let x_i^i be the estimate of the shortest distance from node i to node 1 available at node i. Consider a distributed algorithm whereby each node $i=1,\ldots,n$ executes the iteration

$$x_i^i \leftarrow \min_{j\in N(i)} \{a_{ij}+x_j^j\} \tag{3.1}$$

after receiving one or more estimates x_j^j from its neighbors, while node 1 sets

$$x_1^1 = 0.$$

This algorithm--a distributed asynchronous implementation of Bellman's shortest path algorithm--was implemented on the ARPANET in 1969 [14]. The estimate x_i^i can be shown to converge to the unique shortest distance from node i to node 1 provided the starting values x_i^i are nonnegative [12]. The algorithm clearly is a special case of the model of the previous section. Here the measurement equation [cf. (2.3)] is

$$z_j^i = x_j^j, \qquad \forall\, j\in N(i) \tag{3.2}$$

the measurement update equation [cf. (2.1)] replaces x_j^i by z_j^i and leaves all other coordinates x_m^i, $m\neq j$ unchanged, while the corresponding update formula of (2.2) can be easily constructed using (3.1).

Example 2: (Fixed point calculations)

The preceding example is a speci case of a distributed dynamic programming algorithm (see [12]) which is itself a special case of a distributed fixed point algorithm. Suppose we are interested in computing a fixed point of a mapping $F: X \to X$. We construct a distributed fixed point algorithm that is a special case of the model of the previous section as follows:

Let X be a Cartesian product of the form $X=X_1 x X_2 x \ldots x X_n$ and let us write accordingly $x=(x_1,x_2,\ldots,x_n)$ and $F(x) = (F_1(x),F_2(x),\ldots,F_n(x))$ where $F_i: X \to X_i$. Let $x^i=(x_1^i,\ldots,x_n^i)$ be the estimate of x generated at the ith processor. Processor i executes the iteration.

$$x_j^i \leftarrow \begin{cases} x_j^i & \text{if } i \neq j \\ F_i(x^i) & \text{if } i=j \ , \end{cases} \tag{3.3}$$

(this corresponds to the mapping C_i of (2.2)), and transmits from time to time x_i^i to the other processors. Thus the measurements z_j^i are given by [cf. (2.3)]

$$z_j^i = x_j^j \ , \quad i \neq j \tag{3.4}$$

and the (i,j)th measurement update equation [cf. (2.1)] is given by

$$x_m^i \leftarrow \begin{cases} x_m^i & \text{if } m \neq j \\ z_j^i & \text{if } m=j \ . \end{cases} \tag{3.5}$$

Conditions under which the estimate x^i converges to a fixed point of F are given in [13] (see also Section 4).

Example 3: (Distributed deterministic gradient algorithm)

This example is a special case of the preceding one whereby $X = R^n$, $X_i = R$, and F is of the form

$$F(x) = x - \alpha \nabla f(x) \tag{3.6}$$

where ∇f is the gradient of a function $f: R^n \to R$, and α is a positive scalar stepsize. Iteration (3.3) can then be written as

$$x_j^i \leftarrow \begin{cases} x_j^i & \text{if } i \neq j \\ x_i^i - \alpha \dfrac{\partial f(x^i)}{\partial x_i} & \text{if } i = j \end{cases} \tag{3.7}$$

A variation of this example is obtained if we assume that, instead of each processor i transmitting directly his current value of the coordinate x_i to the other processors, there is a measurement device that transmits the current value of the partial derivative $\dfrac{\partial f(x)}{\partial x_i}$ to the ith processor. In this case there is only one type of measurement for each processor i [cf. (2.3)] and it is given by

$$z_1^i = \frac{\partial f(x_1^1, \ldots, x_n^n)}{\partial x_i} .$$

While the equation above assumes no noise in the measurement of each partial derivative one could also consider the situation where this measurement is corrupted by additive or multiplicative noise thereby obtaining a model of a distributed stochastic gradient method. Many other descent algorithms admit a similar distributed version.

Example 4: (An Organizational Model)

This example is a variation of the previous one, but may be also viewed as a model of collective decision making in a large organization. Let

$X = X_1 x X_2 x \ldots x X_n$ be the feasible set, where X_i is a Euclidean space and let $f: X \to [0,\infty)$ be a cost function of the form $f(x) = \sum_{i=1}^{n} f^i(x)$. We interpret f^i as the cost facing the i-th division of an organization. This division is under the authority of decision maker (processor) i, who updates the i-th component $x_i \in X_i$ of the decision vector x. We allow the cost f^i to depend on the decisions x_j of the remaining decision makers, but we assume that this dependence is weak. That is, let

$$K^i_{jm} = \sup_{x \in X} \left| \frac{\partial^2 f^i(x)}{\partial x_j \partial x_m} \right|$$

and we are interested in the case $K^i_{jm} << K^i_{ii}$ (unless j=m=i). Decision maker i receives measurements z^i_j, j=1,...,n of the form

$$z^i_j(t) = \frac{\partial f^j}{\partial x_i}\left(x^1_1(\tau^{i1}_j(t)), x^2_2(\tau^{i2}_j(t)), \ldots, x^n_n(\tau^{in}_j(t))\right) , \tag{3.8}$$

where $\tau^{ik}_j(t) \leq t$ [cf. (2.3)]. Once in a while, he also updates his decision according to

$$x^i_i(t+1) = x^i_i(t) - \alpha_i \sum_{j=1}^{n} z^i_j(t) . \tag{3.9}$$

If we assume that

$$\tau^{im}_j(t) \leq \tau^{ij}_j(t), \qquad \forall i,j,m,t,$$

the above algorithm admits the following interpretation: each decision maker m, at time $\tau_j^{im}(t)$ sends a message $x_m^m(\tau_j^{im}(t))$, to inform decison maker j of his decision. These messages are the last such messages received by decision maker j no later than $\tau_j^{ij}(t)$. Then, decision maker j (who is assumed to be knowledgeable about f^j) computes z_j^i according to (3.8) and sends it to decision maker i; the latter message is the last such message received by decision maker i no later than t, and is being used, at time t, by decision maker i, to update his decision according to (3.9). On an abstract level, each decision maker j is being informed about the decision of the others and replies by saying how he is affected by their decisions; however, this may be done in an asynchronous and very irregular manner.

Example 5: (Distributed optimal routing in data networks)

A standard model of optimal routing in data networks (see e.g. the survey [6]) involves the multicommodity flow problem

$$\text{minimize} \sum_{a\in A} D_a(F_a)$$

$$\text{subject to } F_a = \sum_{w\in W} \sum_{\substack{p\in P_w \\ a\in p}} x_p, \quad \forall a\in A$$

$$\sum_{p\in P_w} x_p = r_w, \quad \forall w\in W$$

$$x_p \geq 0, \quad \forall w\in W,\ p\in P_w .$$

Here A is the set of directed links in a data network, F_a is the communication rate (say in bits/sec) on link $a\in A$, W is a set of origin-destination

(OD) pairs, P_w is a given set of directed paths joining the origin and the destination of OD pair w, x_p, $p \in P_w$ is the communication rate on path p of OD pair w, r_w is a given required communication rate of OD pair w, and D_a is a monotonically increasing differentiable convex function for each $a \in S$. The objective here is to distribute the required rates r_w among the available paths in P_w so as to minimize a measure of average delay per message as expressed by $\sum_{a \in A} D_a(F_a)$.

Since the origin node of each OD pair w has control over the rates x_p, $p \in P_w$ it is convenient to use a distributed algorithm of the gradient projection type (see [6],[8]) whereby each origin iterates on its own path rates asynchronously and independently of other origins. This type of iteration requires knowledge of the first partial derivatives $D_a'(F_a)$ for each link evaluated at the current link rates F_a. A practical scheme similar to the one currently adopted on the ARPANET [9] is for each link $a \in A$ to broadcast to all the nodes the current value of either F_a or $D_a'(F_a)$. This information is then incorporated in the gradient projection iteration of the origin nodes. In this scheme each origin node can be viewed as a processor and F_a or $D_a'(F_a)$ plays the role of a measurement which depends on the solution estimates of all processors [cf. (2.3)].

The direct opposite of a specialized process, in terms of division of labor between processors, is a totally overlapping process.

Example 6: (Total Overlap)

Let the feasible set X be a Euclidean space. Each processor i receives measurements z_j^i $(j \neq i)$ which are the values of the estimates x^j of other processors; that is,

$$z_j^i = x^j , \qquad i \neq j .$$

Whenever such a measurement is received, processor i updates his estimate by taking a convex combination:

$$x^i \leftarrow M_{ij}(x^i, z_j^i) = \beta^{ij} x^i + (1-\beta^{ij})\, z_j^i \tag{3.10}$$

where $0<\beta^{ij}<1$. Also processor i receives his own information z_i^i, generated according to

$$z_i^i = \phi_i(x^i, \omega)$$

and updates x^i according to

$$x^i(t+1) = M_{ii}(x^i(t),\, z_i^i(t)) = x^i(t) - \alpha^i z_i^i(t) \tag{3.11}$$

where α^i is a positive scalar stepsize.† Such an algorithm is of interest if the objective is to minimize a cost function $f: X \mapsto R$, and $z_i^i(t)$ is in some sense a descent direction with respect to f. In a deterministic setting, such a scheme could be redundant, as some processors would be close to replicating the computation of others. In a stochastic setting, however (e.g. if

$$z_i^i(t) = \frac{\partial f}{\partial x}(x^i(t)) + w^i(t),$$

where $w^i(t)$ is zero-mean white noise) the combining process is effectively averaging out the effects of the noise and may improve convergence .

Example 7: (System Identification)

Consider two moving average processes $y^1(t), y^2(t)$ generated according to

$$y^i(t) = A(q)u(t) + w^i(t),$$

---

†The stepsize α^i could be constant as in deterministic gradient methods. However, in other cases (such as stochastic gradient methods with additive noise) it is essential that α^i is time varying and tends to zero. This, strictly speaking, violates the assumption that the mapping M_{ij} does not depend on the time t. However it is possible to circumvent this by introducing (as an additional component of x^i) a local counter at each processor i that keeps track of the number of times iteration (3.10) or (3.11) is executed at processor i. The stepsize α^i could be made dependent on the value of this local counter (see the discussion following (2.1) and (2.2) in Section 2).

where $A(\cdot)$ is a polynomial, to be identified, q is the unit delay operator and $w^i(t)$, i=1,2, are white, zero-mean processes, possibly correlated with each other. Let there be two processors (n=2); processor i measures $y^i(t)$ and both measure u(t) at each time t. Each processor i updates his estimate x^i of the coefficients of A according to any of the standard system identification algorithms (e.g. the LMS or RLS algorithm). Under the usual identifiability conditions [33] each processor would be able to identify $A(\cdot)$ by himself. However, convergence should be faster if once in a while one processor gets (possibly delayed) measurements of the estimates of the other processor and combines them by taking a convex combination. Clearly, this is a special case of Example 6.

A more complex situation arises if we have two ARMAX processes y^1, y^2, driven by a common colored noise:

$$A^i(q)y^i(t) = B^i(q)u^i(t) + w(t), \quad i=1,2$$
$$w(t) = C(q)v(t),$$

where v(t) is white and A^i, B^i, C are polynomials in the delay operator q. Assuming that each processor i observes y^i and u^i, he may under certain conditions [34] identify A^i, B^i. In doing this he must, however, identify the common noise source C as well. So we may envisage a scheme whereby processors use a standard algorithm to identify A^i, B^i, C and once in a while exchange messages with their estimates of the coefficients of C; these estimates are then combined by taking a convex combination.

This latter example falls in between the extreme cases of specialization and total overlap: there is specialization concerning the coefficients of A^i, B^i and overlap concerning the coefficients of C.

4. Convergence of Contracting Processes

In our effort to develop a general convergence result for the distributed algorithmic model of Section 2 we draw motivation from existing convergence theories for (centralized) iterative algorithms. There are several theories of this type (Zangwill [15], Luenberger [16], Ortega and Rheinboldt [17]--the most general are due to Poljak [18] and Polak [19]). Most of these theories have their origin in Lyapunov's stability theory for differential and difference equations. The main idea is to consider a generalized distance function (or Lyapunov function) of the typical iterate to the solution set. In optimization methods the objective function is often suitable for this purpose while in equation solving methods a norm of the difference between the current iterate and the solution is usually employed. The idea is typically to show that at each iteration the value of the distance function is reduced and reaches its minimum value in the limit.

The result of this section is based on a similar idea. However instead of working with a generalized distance function we prefer to work (essentially) with the level sets of such a function; and instead of working with a single processor iterate (as in centralized processes) we work with what may be viewed as a state of computation of the distributed process which includes all current processor iterates and all latest information available at the processors.

The subsequent result is reminiscent of convergence results for successive approximation methods associated with contraction mappings. For this reason we refer to processes satisfying the following assumption as contracting processes. In what follows in this section we assume that the feasible set X in the model of Section 2 is a topological space so we can talk about convergence of sequences in X.

Assumption 3.1: There exists a sequence of sets X(k) with the following properties:

a) $X^* \subset X(k+1) \subset X(k) \subset \ldots \subset X$

b) If $\{x_k\}$ is a sequence in X such that $x_k \in X(k)$ for all k, then $\{x_k\}$ converges to a solution.

(Note: If the notion of sequence convergence to a subset is defined on X, one may replace convergence of $\{x_k\}$ to a solution with convergence to the solution set X^*).

c) For all i, j and k denote:

$$Z_j^i(k) = \{\phi_{ij}(x^1,\ldots,x^n,\omega) \mid x^1 \in X(k) \ldots, x^n \in X(k),\ \omega \in \Omega\} \tag{4.1}$$

$$\bar{X}^i(k) = \{C_i(x^i, z_1^i, \ldots, z_{m_i}^i) \mid x^i \in X(k),\ z_1^i \in Z_1^i(k), \ldots, z_{m_i}^i \in Z_{m_i}^i(k)\} \tag{4.2}$$

$$\bar{Z}_j^i(k) = \{\phi_{ij}(x^1,\ldots,x^n,\omega) \mid x^1 \in \bar{X}^1(k), \ldots, x^n \in \bar{X}^n(k),\ \omega \in \Omega\} \tag{4.3}$$

The sets X(k) and the mappings ϕ_{ij}, M_{ij}, and C_i are such that for all i,j and k

$$\bar{X}^i(k) \subset X(k) \tag{4.4}$$

$$M_{ij}(x^i, z_1^i, \ldots, z_{m_i}^i) \in X(k),\ \forall x^i \in X(k),\ z_1^i \in Z_1^i(k), \ldots, z_{m_i}^i \in Z_{m_i}^i(k) \tag{4.5}$$

$$M_{ij}(x^i, z_1^i, \ldots, z_{m_i}^i) \in \bar{X}^i(k),\ \forall x^i \in \bar{X}^i(k),\ z_1^i \in Z_1^i(k), \ldots, z_{m_i}^i \in Z_{m_i}^i(k) \tag{4.6}$$

$$M_{ij}(x^i, z_1^i, \ldots, z_{m_i}^i) \in X(k+1),\ \forall x^i \in \bar{X}^i(k),\ z_1^i \in \bar{Z}_1^i(k), \ldots, z_{m_i}^i \in \bar{Z}_{m_i}^i(k)\ . \tag{4.7}$$

Assumption 3.1 is a generalized version of a similar assumption in reference [13]. Broad classes of deterministic specialized processes satisfying the assumption are given in that reference. The main idea is that membership in the set X(k) is

representative in some sense of the proximity of a processor estimate to a solution. By part b), if we can show that a processor estimate successively moves from $X(0)$ to $X(1)$, then to $X(2)$ and so on, then convergence to a solution is guaranteed. Part c) assures us that once all processor estimates enter the set $X(k)$ then they remain in the set $X(k)$ [cf. (4.4),(4.5)] and (assuming all processors keep on computing and receiving measurements) eventually enter the set $X(k+1)$ [cf. (4.6),(4.7)]. In view of these remarks the proof of the following result is rather easy. Note that the assumption does not differentiate the effects of two different members of the probability space [cf. part c)] so it applies to situations where the process is either deterministic (Ω consists of a single element), or else stochastic variations are not sufficiently pronounced to affect the membership relations in part c).

Proposition 3.1: Let Assumptions 2.1, 2.2, 3.1, hold and assume that all initial processor estimates x^i, $i=1,\ldots,n$ belong to $X(0)$, while all initial measurements z_j^i available at the processors belong to the corresponding sets $Z_j^i(0)$. Then each of of the sequence $\{x^i\}$ converges almost surely to a solution as $t\to\infty$.

The proof will not be given since it is very similar to the one given in [13]. Note that the proposition does not guarantee asymptotic agreement of the processor estimates but in situations where Assumption 3.1 is satisfied one can typically also show agreement.

Example 2 (continued): As an illustration consider the specialized process for computing a fixed point of a mapping F in example 2. There X is a Cartesian product $X_1 \times X_2 \times \ldots \times X_n$, and each processor i is responsible for updating the

ith "coordinate" x_i of $x = (x_1, x_2, \ldots, x_n)$ while relying on essentially direct communications from other processors to obtain estimates of the other coordinates. Suppose that each set X_i is a Banach space with norm $||\cdot||_i$ and X is endowed with the sup norm

$$||x|| = \max\{||x_1||_1, \ldots, ||x_n||_n\}, \qquad \forall x \in X \tag{4.8}$$

Assume further that F is a contraction mapping with respect to this norm, i.e., for some $\alpha \in (0,1)$.

$$||F(x)-F(y)|| \leq \alpha||x-y||, \qquad \forall x,y \in X . \tag{4.9}$$

Then the solution set consists of the unique fixed point x^* of F. For some positive constant B let us consider the sequence of sets

$$X(k) = \{x \in X \mid ||x-x^*|| \leq B\alpha^k\} , \qquad k=0,1,\ldots,$$

The sets defined by (4.1)-(4.3) are then given by

$$Z_j^i(k) = \{x_j \in X_j \mid ||x_j - x_j^*||_j \leq B\alpha^k\}$$

$$\overline{X}^i(k) = \{x \in X(k) \mid ||x_i - x_i^*||_i \leq B\alpha^{k+1}\}$$

$$\overline{Z}_j^i(k) = \{x_j \in X_j \mid ||x_j - x_j^*||_j \leq B\alpha^{k+1}\} .$$

It is straightforward to show that the sequence {X(k)} satisfies Assumption 3.1. Further illustrations related to this example are given in [13]. Note however that the use of the sup norm (4.8) is essential for the verification of Assumption 3.

Similarly Assumption 3 can be verified in the preceding example if the contraction assumption (4.9) is substituted by a monotonicity assumption (see [13]). This monotonicity assumption is satisfied by most of the dynamic programming problems of interest including the shortest path problem of example 1 (see also [12]). An important exception is the infinite horizon average cost Markovian decision problem (see [12], p. 616).

An important special case for which the contraction mapping assumption (4.9) is satisfied arises when $X=R^n$ and $x_1,x_2,\ldots,x_n$ are the coordinates of x. Suppose that F satisfies

$$|F(x)-F(y)| \leq P|x-y|, \qquad \forall x,y \in R^n \tag{4.10}$$

where P is an nxn matrix with nonnegative elements and spectral radius strictly less than unity, and for any $z=(z_1,z_2,\ldots,z_n)$ we denote by $|z|$ the column vector with coordinates $|z_1|,|z_2|,\ldots,|z_n|$. Then F is called a P-contraction mapping. Fixed point problems involving such mappings arise in dynamic programming ([20], p.374), and solution of systems of nonlinear equations ([17], Section 13.1). It can been shown ([11], p.231) that if F is a P-contraction then it is a contraction mapping with respect to some norm of the form (4.8). Therefore Proposition 3.1 applies.

We finally note that it is possible to use Proposition 3.1 to show convergence of similar fixed point distributed processes involving partial or total overlaps between the processors (compare with example 6).

Example 3 (continued): Consider the special case of the deterministic gradient algorithm of example 3 corresponding to the mapping

$$F(x) = x - \alpha \nabla f(x) \ . \tag{4.11}$$

Assume that $f:R^n \to R$ is a twice continuously differentiable convex function with Hessian matrix $\nabla^2 f(x)$ which is positive definite for all x. Assume also that there exists a unique minimizing point x* of f over R^n. Consider the matrix

$$H^* = \begin{bmatrix} \left|\frac{\partial^2 f}{(\partial x_1)^2}\right| & -\left|\frac{\partial^2 f}{\partial x_1 \partial x_2}\right| & \cdots & -\left|\frac{\partial^2 f}{\partial x_1 \partial x_n}\right| \\ \vdots & \vdots & & \vdots \\ -\left|\frac{\partial^2 f}{\partial x_n \partial x_1}\right| & -\left|\frac{\partial^2 f}{\partial x_n \partial x_2}\right| & \cdots & \left|\frac{\partial^2 f}{(\partial x_n)^2}\right| \end{bmatrix} \tag{4.12}$$

obtained from the Hessian matrix $\nabla^2 f(x^*)$ by replacing the off-diagonal terms by their negative absolute values. It is shown in [13] that if the matrix H^* is positive definite then the mapping F of (4.11) is a P-contraction within some open sphere centered at x^* provided the stepsize α in (4.11) is sufficiently small. Under these circumstances the distributed asynchronous gradient method of this example is convergent to x^* provided all initial processor estimates are sufficiently close to x^* and the stepsize α is sufficiently small. The neighborhood of local convergence will be larger if the matrix (4.12) is positive definite within an accordingly larger neighborhood of x^*. For example if f is positive definite quadratic with the corresponding matrix (4.12) positive definite a global convergence result can be shown.

One condition that guarantees that H^* is positive definite is <u>strict diagonal dominance</u> ([17], p.48-51).

$$\frac{\partial^2 f}{(\partial x_i)^2} > \sum_{\substack{j=1 \\ j\neq i}}^{n} \left| \frac{\partial^2 f}{\partial x_i \partial x_j} \right| , \qquad \forall\, i=1,\ldots,n,$$

where the derivatives above are evaluated at x^*. This type of condition is typically associated with situations where the coordinates of x are <u>weakly coupled</u> in the sense that changes in one coordinate have small effects on the first partial derivatives of f with respect to the other coordinates. This result can be generalized to the case of <u>weakly coupled systems</u> (as opposed to weakly coupled coordinates). Assume that x is partitioned as $x=(x_1,x_2,\ldots,x_n)$ where now $x_i \in R^{m_i}$ (m_i may be greater than one but all other assumptions made earlier regarding f are in effect). Assume that there are n processors and the ith processor asynchronously updates the subvector x_i according to an approximate form of Newton's method where the second

order submatrices of the Hessian $\nabla^2_{x_i x_j} f$, $i \neq j$ are neglected, i.e.

$$x_i \leftarrow x_i - (\nabla^2_{x_i x_i} f)^{-1} \nabla_{x_i} f \ . \tag{4.13}$$

In calculating the partial derivatives above processor i uses the values x_j latest communicated from the other processors $j \neq i$ similarly as in the distributed gradient method. It can be shown that if the cross-Hessians $\nabla^2_{x_i x_j} f$, $i \neq j$ have sufficiently small norm relative to $\nabla^2_{x_i x_i} f$, then the totally asynchronous version of the approximate Newton method (4.13) converges to x^* if all initial processor estimates are sufficiently close to x^*. The same type of result may also be shown if (4.13) is replaced by

$$x_i \leftarrow \arg \min_{\substack{m_i \\ x_i \in R}} f(x_1, x_2, \ldots, x_n) \ . \tag{4.14}$$

Unfortunately it is not true always that the matrix (4.12) is positive definite, and there are problems where the totally asynchronous version of the distributed gradient method is not guaranteed to converge regardless of how small the stepsize α is chosen. As an example consider the function $f: R^3 \rightarrow R$

$$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3)^2 + (x_1 + x_2 + x_3 - 3)^2 + \varepsilon(x_1^2 + x_2^2 + x_3^2)$$

where $0 < \varepsilon << 1$. The optimal solution is close to $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ for ε: small. The scalar ε plays no essential role in this example. It is introduced merely for the purpose of making the Hessian of f positive definite. Assume that all initial processor estimates are equal to some common value $\bar{x}$, and that processors execute many gradient

iterations with a small stepsize before communicating the current values of their respective coordinates to other processors. Then (neglecting the terms that depend on ϵ) the ith processor tries in effect to solve the problem

$$\min_{x_i} \{(x_i+2\bar{x})^2 + (x_i+2\bar{x}-3)^2\}$$

thereby obtaining a value close to $\frac{3}{2} - 2\bar{x}$. After the processor estimates of the respective coordinates are exchanged each processor coordinate will have been updated approximately according to

$$\bar{x} \leftarrow \frac{3}{2} - 2\bar{x} \tag{4.15}$$

and the process will be repeated. Since (4.15) is a divergent iterative process we see that, regardless of the stepsize chosen and the proximity of the initial processor estimates to the optimal solution, by choosing the communication delays sufficiently large the distributed gradient method can be made to diverge when the matrix H^* of (4.12) is not positive definite.

5. Convergence of Descent Processes

We saw in the last section that the distributed gradient algorithm converges appropriately when the matrix (4.12) is positive definite. This assumption is not always satisfied, but convergence can be still shown (for a far wider class of algorithms) if a few additional conditions are imposed on the frequency of obtaining measurements and on the magnitude of the delays in equation (2.3). The main idea behind the results described in this section is that if delays are not too large, if certain processors do not obtain measurements and do not update much more frequently than others, then the effects of asynchronism are relatively small and the algorithm behaves approximately as a centralized algorithm, similar to the class of centralized pseudo-gradient algorithms considered in [40].

Let $X = X_1 x X_2 x \ldots x X_L$ be the feasible set, where X_ℓ ($\ell=1,\ldots,L$) is a Banach space. If $x=(x_1,\ldots,x_L)$, $x_\ell \in X_\ell$, we refer to x_ℓ as the ℓ-th component of x. We endow X with the sup norm, as in (4.8). Let $f:x\to[0,\infty)$ be a cost function to be minimized. We assume that f is Frechet differentiable and its derivative is Lipschitz continuous.

Each processor i keeps in his memory an estimate $x^i(t) = (x_1^i(t),\ldots,x_L^i(t)) \in X$ and receives measurements $z_{j,\ell}^i \in X_\ell$, $i\neq j$, with the value of the ℓ-th component of x^j, evaluated by processor j at some earlier time $\tau_{j,\ell}^i(t) \leq t$; that is, $z_{j,\ell}^i(t) = x_\ell^j(\tau_{j,\ell}^i(t))$. He also receives from the environment exogenous, possibly stochastic measurements $z_i^i \in X$, which are in a direction of descent with respect to the cost function f, in a sense to be made precise later. We denote by $z_{i,\ell}^i$ the ℓ-th component of z_i^i.

Whenever processor i receives measurements $z_{j,\ell}^i$, he updates his estimate vector x^i componentwise, according to:

$$x_\ell^i(t+1) = \beta_{i,\ell}^i(t)x_\ell^i(t) + \sum_{j\neq i} \beta_{j,\ell}^i(t)z_{j,\ell}^i(t) + \alpha^i(t)z_{i,\ell}^i(t) . \qquad (5.1)$$

The coefficients $\beta_{j,\ell}^i(t)$ are nonnegative scalars satisfying

$$\sum_{j=1}^n \beta_{j,\ell}^i(t)=1, \qquad \forall i,\ell,t,$$

and such that: if no measurement $z_{j,\ell}^i$ was received by processor i ($i\neq j$) at time t, then $\beta_{j,\ell}^i(t)=0$. That is, processor i combines his estimate of the ℓ-th component of the solution with the estimates (possibly outdated) of other processors that he has just received, by forming a convex combination. Also, if no new measurement z_i^i was obtained at time t, we should set $z_i^i(t)=0$ in equation (5.1). The coefficient

$\alpha^i(t)$ is a nonnegative stepsize. It can be either independent of t or it may depend on the number of times up to t that a new measurement (of any type) was received at processor i.

Equation (5.1) which essentially defines the algorithm, is a linear system driven by the exogenous measurements $z_i^i(t)$. Therefore, there exist linear operators $\Phi^{ij}(t|s),(t\geq s)$, such that

$$x^i(t) = \sum_{j=1}^{n} \Phi^{ij}(t|0)x^j(1) + \sum_{s=1}^{t-1} \sum_{j=1}^{n} \alpha^j(s)\Phi^{ij}(t|s)z_j^j(s) .$$

We now impose an assumption which states that if the processors cease obtaining exogenous measurements from some time on (that is, if they set $z_i^i=0$), they will asymptotically agree on a common limit:

Assumption 5.1: For any i,j,s, $\lim_{t\to\infty} \Phi^{ij}(t|s)$ exists (with respect to the induced operator norm) and is the same for all i. The common limit is denoted by $\Phi^j(s)$.

Assumption 5.1 is very weak. Roughly speaking it requires that for every component $\ell\in\{1,\ldots,L\}$ there exists a directed graph $G=(N,A)$, where the set N of nodes is the set $\{1,\ldots,n\}$ of processors, and such that there exists a path from every processor to every other processor. Also the coefficients $\beta_{i,\ell}^j(t)$ must be such that "sufficient combining" takes place and the processors tend to agree.

We can now define a vector $y(t)\in X$ by

$$y(t) = \sum_{j=1}^{n} \Phi^j(0)x^j(1) + \sum_{s=1}^{t-1} \sum_{j=1}^{n} \alpha^j(s)\Phi^j(s)z_j^j(s)$$

and observe that $y(t)$ is recursively generated by

$$y(t+1) = y(t) + \sum_{i=1}^{n} \alpha^i(t)\phi^i(t)z_i^i(t) \ . \tag{5.2}$$

We can now explain the main idea behind the results to be described: if $\phi^{ij}(t|s)$ converges to $\phi^j(s)$ fast enough, if $\alpha^i(t)$ is small enough, and if $z_i^i(t)$ is not too large, then $x^i(t)$, for each i, will evolve approximately as $y(t)$. We may then study the behavior of the recursion (5.2) and make inferences about the behavior of $x^i(t)$.

The above framework covers both specialized processes, in which case we have L=n, as well as the case of total overlap where we have L=1 and we do not distinguish between components of the estimates. For specialized processes (e.g. example 3) it is easy to see that $y(t) = (x_1^1(t), x_2^2(t), \ldots, x_n^n(t))$.

We now proceed to present some general convergence results. We allow the exogenous measurements z_i^i of each processor, as well as the initialization $x^i(1)$ of the algorithm to be random (with finite variance). We assume that they are all defined on a probability space (Ω, F, P) and we denote by F_t the σ-algebra generated by $\{x^i(1), z_i^i(s);\ i=1,\ldots,n;\ s=1,\ldots,t-1\}$. We assume, however, that the sequence of times at which measurements are obtained, computations are performed, the times $\tau_{j,\ell}^i(t)$, as well as the combining coefficients $\beta_{j,\ell}^i(t)$ are deterministic. (In fact, this assumption may be often relaxed). In order to quantify the speed of convergence of $\phi^{ij}(t|s)$ we introduce

$$c(t|s) = \max_{i,j} |\phi^{ij}(t|s) - \phi^j(s)| \ .$$

By Assumption 5.1 $\lim_{t\to\infty} c(t|s)=0$ and it may be shown that $c(t|s) \leq 1$, $\forall t,s$. Consider the following assumptions:

Assumption 5.2:

$$E\left[\left\langle \frac{\partial f}{\partial x}\,(x^i(t)),\ \phi^i(t)z_i^i(t)\right\rangle \,|F_t\right]\leq 0, \qquad \forall\, t,i, \quad \text{a.s.}$$

Assumption 5.3:

a) For some $K_o \geq 0$

$$E\left[\,||z_i^i(t)||^2\right] \leq -\,K_o E\left[\left\langle \frac{\partial f}{\partial x}\,(x^i(t)), \phi^i(t)z_i^i(t)\right\rangle\right], \qquad \forall\, i,t\ .$$

b) For some $B\geq 0$, $d\in[0,1)$, $\epsilon(t|s)\leq Bd^{t-s}$, $\forall t\geq s$, $\forall s$.

Assumption 5.2 states that $\phi^i(t)z_i^i(t)$ (which is the "effective update direction" of processor i, see (5.2)) is a descent direction with respect to f. Assumption 5.3a requires that $z_i^i(t)$ is not too large. In particular any noise present in $z_i^i(t)$ can only be "multiplicative-like": its variance must decrease to zero as a stationary point of f is approached. For example, we may have

$$z_i^i(t) = -\left[\frac{\partial f}{\partial x}\,(x^i(t))\right](1+w^i(t)),$$

where $w^i(t)$ is scalar white noise. Finally, Assumption 5.3b requires that the processors tend to agree exponentially fast. Effectively, this requires that the time between consecutive measurements of the type $z_{j,\ell}^i$, $i\neq j$, as well as the delays $t-\tau_{j,\ell}^i(t)$ are bounded together with some minor restriction of the coefficients $\beta_{j,\ell}^i(t)$ for those times that a measurement of type $z_{j,\ell}^i$ is obtained.

Letting

$$\alpha_o = \sup_{t,i} \alpha^i(t) ,$$

we may use Assumptions 5.3a and 5.3b to show that $||x^i(t)-y(t)||$ is of the order of α_o. Using the Lipschitz continuity of $\frac{\partial f}{\partial x}$ it follows that $||\frac{\partial f}{\partial x}(y(t)) - \frac{\partial f}{\partial x}(x^i(t))||$ is also of the order of α_o; then, using Assumption 5.2, it follows that (5.2) corresponds to a descent algorithm, up to first order in α_o. Choosing α_o small enough, convergence may be shown by invoking the supermartingale convergence theorem. The above argument can be made rigorous and yields the following proposition (the proof may be found in [35] and in a forthcoming publication):

Proposition 5.1: If Assumptions 5.1, 5.2, 5.3, hold and if α_o is small enough, then:

a) $f(x^i(t))$, $i=1,\ldots,n$, *as well as* $f(y(t))$ *converge*, almost surely, and to the same limit.

b) $\lim_{t\to\infty} (x^i(t)-y(t))=0$, $\forall i$, almost surely and in the mean square.

c) $$\sum_{t=1}^{\infty} \sum_{i=1}^{n} \alpha^i(t) E\left[< \frac{\partial f}{\partial x}(x^i(t)), \Phi^i(t) z_i^i(t) > | F_t \right] > -\infty , \qquad (5.3)$$

almost surely. The expectation of the above expression is also finite.

A related class of algorithms arises if the noise in $z_i^i(t)$ is allowed to be additive, e.g.

$$z_i^i(t) = \frac{\partial f}{\partial x}(x^i(t)) + w^i(t) ,$$

where $w^i(t)$ is zero-mean white. In such a case, an algorithm may be convergent only if $\lim_{t\to\infty} \alpha^i(t)=0$. In fact, $\alpha^i(t)=1/t_i$, where t_i is the number of times up to time t that a new measurement was received at i, is the most convenient choice, and this is what we assume here. However, this choice of stepsize implies that the algorithm becomes progressively slower, as $t\to\infty$. We may therefore allow the agreement process to become progressively slower as well, and still retain convergence. In physical terms, the time between consecutive measurements $z^i_{j,\ell}(i\neq j)$ may increase to infinity, as $t\to\infty$. In mathematical terms:

Assumption 5.4: a) For some K_o, K_1, $K_2 \geq 0$,

$$E\left[\left|\left|z_i^i(t)\right|\right|^2\right] \leq -K_o E\left[\left\langle \frac{\partial f}{\partial x}(x^i(t)), \Phi^i(t) z_i^i(t)\right\rangle\right] + K_1 E\left[f(x^i(t))\right] + K_2$$

b) For some $B\geq 0$, $\delta\in(0,1]$, $d\in[0,1)$

$$c(t|s) \leq B d^{t^\delta - s^\delta}, \quad \forall t\geq s, \forall s.$$

We then have [35]:

Proposition 5.2: Let $\alpha^i(t)=1/t_i$, where t_i is the number of times up to time t that a new measurement was received at i, and assume that for some $\varepsilon>0$, $t_i\geq\varepsilon\cdot t$ for all i,t. Asumme also that Assumptions 5.1, 5.2, 5.4 hold. Then the conclusions (a),(b),(c) of Proposition 5.1 remain valid.

Propositions 5.1,5.2 do not prove yet convergence to the optimum (suppose, for example, that $z_i^i(t)\equiv 0$, $\forall i,t$). However, (5.3) may be exploited to yield optimality under a few additional assumptions:

Corollary: Let the assumptions of either Proposition 5.1 or 5.2 hold. Let T^i be the set of times that processor i obtains a measurement of type z_i^i. Suppose that there exists some $B \geq 0$ and, for each i, a sequence $\{t_k^i\}$ of distinct elements of T^i such that

$$\max_{i,j} |t_k^i - t_k^j| \leq B , \tag{5.4}$$

$$\sum_{k=1}^{\infty} \min_i \{\alpha^i(t_k^i)\} = \infty .$$

Finally, assume that there exist uniformly continuous functions: $g^i : x \to [0,\infty)$ satisfying

a) $$\lim_{|x| \to \infty} \inf \sum_{i=1}^{n} g^i(x) > 0$$

b) $$E\left[\left\langle \frac{\partial J}{\partial x}(x^i(t)), \Phi^i(t) z_i^i(t) \right\rangle | F_t \right] \leq - g^i(x^i(t)), \quad \forall t \in T^i, \forall i, \text{ almost surely.}$$

c) $$\sum_{i=1}^{n} g^i(x^*) = 0 \Rightarrow x^* \in X^* \triangleq \{x \in X \mid f(x^*) = \inf_x f(x)\}$$

Then, $\lim_{t \to \infty} f(x^i(t)) = \inf_x f(x)$, $\forall i$, almost surely.

Example 3: (continued): It follows from the above results that the distributed deterministic gradient algorithm applied to a convex function converges provided that a) The stepsize α is small enough, b) Assumption 5.3(b) holds and c) The processors update, using (3.7), regularly enough, i.e. condition (5.4) is satisfied. Similarly, convergence for the distributed stochastic gradient algorithm follows if we choose a stepsize $\alpha^i(t) = 1/t_i$, if Assumption 5.4 and condition (5.4) hold.

Example 4: (continued) Similarly with the previous example, convergence to stationary points of f may be shown, provided that α_i is not too large, that the delays $t-\tau_j^{im}(t)$ are not too large and that the processors do not update too irregularly. It should be pointed out that a more refined set of sufficient conditions for convergence may be obtained, which links the "coupling constants" $K_{j,m}^{i}$ with bounds on the delays $t-\tau_j^{im}(t)$ [35]. These conditions effectively quantify the notion that the time between consecutive communications and communication delays between decision makers should be inversely proportional to the strength of coupling between their respective divisions.

Example 7: (continued) Several common algorithms for identification of a moving average process satisfy the conditional descent Assumption 5.2. (e.g. the Least Mean Squares algorithm, or its normalized version-NLMS). Consequently, Proposition 5.2 may be invoked. Using part (c) of the Proposition, assuming that the input is sufficiently rich and that enough messages are exchanged, it follows that the distributed algorithm will correctly identify the system. A detailed analysis is given in [35].

A similar approach may be taken to analyze distributed stochastic algorithms in which the noises are correlated and Assumption 5.2 fails to hold. Very few global convergence results are available even for centralized such algorithms [34,36] and it is an open question whether some distributed versions of them also converge. However, as in the centralized case one may associate an ordinary differential equation with such an algorithm as in [37,38], and prove local convergence subject to an assumption that the algorithm returns infinitely often to a bounded region (see [35]). Such results may be used, for example, to demonstrate local convergence of a distributed extended least squares (ELS) algorithm, applied to the ARMAX identification problem in Example 7.

6. Convergence of Distributed Processes with Bayesian Updates

In Section 4 and 5 we considered distributed processes in which a solution is being successively approximated, while the structure of the updates is restricted to be of a special type. In this section we take a different approach and we assume that the estimate computed by any processor at any given time is such that it minimizes the conditional expectation of a cost function, given the information available to him at that time. Moreover, all processors "know" the structure of the cost function and the underlying statistics, and their performance is only limited by the availability of posterior information. Whenever a processor receives a measurement z_j^i (possibly containing an earlier estimate of another processor) his information changes and a new estimate may be computed.

Formally, let $X = R^m$ be the feasible set, (Ω,F,P) a probability space and $f: X\times\Omega\mapsto[0,\infty)$ a random cost function which is strongly convex in x for each $\omega\in\Omega$. Let $I^i(t)$ denote the information of processor i at time t, which generates a σ-algebra $F_t^i \subset F$. At any time that the information of processor i changes, he updates his estimate according to

$$x^i(t+1) = \arg\min_{x\in X} E[\, f(x,\omega) \,|\, F_t^i] \tag{6.1}$$

Assuming that f is jointly measurable, this defines an almost surely unique, F_t^i - measurable random variable [39].

The information $I^i(t)$ of processor i may change in one of the following ways:

a) New exogenous measurements $z_i^i(t)$ are obtained, so that $I^i(t) = (I^i(t-1), z_i^i(t))$.

b) Measurements $z_j^i(t)$ with the value of an earlier estimate of processor i are obtained; that is,

$$z_j^i(t) = x^j(\tau_j^i(t)); \quad \tau_j^i(t) \leq t$$
$$I^i(t) = (I^i(t-1), z_j^i(t)) \tag{6.2}$$

c) Some information in $I^i(t-1)$ may be "forgotten"; that is, $I^i(t) \subset I^i(t-1)$ (or $F_t^i \subset F_{t-1}^i$).

The times at which measurements are obtained as well as the delays are either deterministic or random; if they are random, their statistics are described by (Ω,F,P) and these statistics are known by all processors.

Case 1: Increasing Information. We start by assuming that information is never forgotten, i.e. $F_{t+1}^i \supset F_t^i$, $\forall i,t$. Let $f(x,\omega) = ||x-x^*(\omega)||^2$, where $x^*:\Omega \to R^m$ is an unknown random vector to be estimated. Then,

$$x^i(t+1) = E[x^*(\omega))|F_t^i]$$

and by the martingale convergence theorem, $x^i(t)$ converges almost surely to a random variable y^i. Moreover it has been shown that if "enough" measurements of type (6.2) are obtained by each processor, then $y^i=y^j$, $\forall i,j$, almost surely [30,41]. If f is not quadratic but strongly convex, the same results are obtained except that convergence holds in the sense of probability and in the $L^2(\Omega,F,\mu)$ sense, where μ is a measure equivalent to P, determined by the function f [39]. However, this scheme is not, strictly speaking, iterative, since $I^i(t)$ increases, and unbounded memory is required.

Case 2: Iterative schemes

The above scheme can be made iterative if we allow processors to forget their past information. For example, let

$$I^i(t) = \begin{cases} \{x^i(t), z_j^i(t)\}, & \text{if a measurement } z_j^i(t) \text{ is obtained at time } t \\ \{x^i(t)\}, & \text{otherwise} \end{cases}$$

Let $z_j^i(t) = x^j(\tau_j^i(t))$, $i \neq j$, $\tau_j^i(t) \leq t$. Assuming that "enough" measurements of this type are obtained by each processor, asymptotic agreement may be still obtained, as for Case 1 [39]. It has been also shown that $x^i(t+1) - x^i(t)$ converges to zero, for each i, but it is not known whether $x^i(t)$ is guaranteed to converge or not.

Even though this case corresponds to an iterative algorithm, it may be very hard to implement: The computation of the minimum in (6.1) may be intractable. Also, even if the processors asymptotically converge and agree, there are no guarantees in general about the quality of the final estimate. There is one notable exception where these drawbacks disappear, which we discuss below:

Case 3: Distributed Linear Estimation

Let $f(x,\omega) = ||x-x^*(\omega)||^2$, where x^* is a zero-mean Gaussian scalar random variable to be estimated. Suppose that at time zero each processor obtains measurements

$$z_{i,k}^i = x^* + w_k^i, \quad k=1,\ldots,m_i \ , \tag{6.3}$$

where w_k^i are zero-mean Gaussian noises. We allow the noises of different processors to be correlated to each other. Let $I^i(0) = \{z_{i,k}^i | k=1,\ldots,m_i\}$. No further measurements of the form (6.3) are obtained after time zero. Subsequently each processor i receives from time to time measurements $z_j^i(t) = z^j(\tau_j^i(t)), \tau_j^i(t) \leq t$, of the other processors' estimates and updates according to

$$x^i(t+1) = E[x^* | I^i(0), z_j^i(t)] \ .$$

The timing and delay of these latter measurements is assumed to be deterministic. If we make the assumption that an infinite number of measurements of each type z_j^i is obtained by each processor i, together with an additional assumption that essentially requires that there exists an indirect communication path between every pair of processors then it can be shown that $x^i(t)$ converges in the mean square to the centralized estimate

$$x^* = E[x^* | I^1(0),\ldots,I^n(0)],$$

which is the optimal estimate of x^* given the total information of all processors [35], [3

What is interesting about the above algorithm is that it corresponds to a distributed iterative decomposition algorithm for solving the centralized linear estimation problem. The minimization of the cost criterion over a space of dimension $\sum_{i=1}^{n} m_i$, in general, is substituted by a sequence of minimizations along (m_i+1)-dimensional subspaces.

If the noises w_k^i, w_ℓ^i, $i \neq j$, are independent the algorithm converges after finitely many iterations. In general, the algorithm converges linearly but the rate of convergence depends strongly on the number of processors and the angles between certain subspaces of random variables (essentially on the correlations between w_k^i and w_ℓ^i, $i \neq j$, see [35],[39]).

REFERENCES

[1] Tannenbaum, A.S., Computer Networks, Prentice Hall, Englewood Cliffs, N.J., 1981.

[2] Schwartz, M., Computer Communication Network Design and Analysis, Prentice Hall, Englewood Cliffs, N.J., 1977.

[3] Kleinrock, L., Queuing Systems, Vol. I & II, J. Wiley, N.Y., 1975.

[4] Schwartz, M., and Stern, T.E., "Routing Techniques Used in Computer Communication Networks," IEEE Trans. on Communications, Vol. COM-28, 1980, pp. 539-552.

[5] Gallager, R.G., "A Minimum Delay Routing Algorithm Using Distributed Computation," IEEE Trans. on Communication, Vol. COM-25, 1977, pp. 73-85.

[6] Bertsekas, D.P., "Optimal Routing and Flow Control Methods for Communication Networks," in Analysis and Optimization of Systems (A. Bensoussan and J.L. Lions, eds.), Springer-Verlag, Berlin and N.Y., 1982, pp. 615-643.

[7] Kung, H.T., "Synchronized and Asynchronous Parallel Algorithms for Multiprocessors," in Algorithms and Complexity, Academic Press, 1976, pp. 153-200.

[8] Bertsekas, D.P., and Gafni, E.M., "Projected Newton Methods and Optimization of Multicommodity Flows," M.I.T., LIDS Report P-1140, M.I.T., Aug. 1981, IEEE Trans. on Aut. Control, Dec. 1983 (to appear).

[9] McQuillan, J.M., Richer, I., and Rosen, E.C., "The New Routing Algorithm for the ARPANET," IEEE Trans. on Communications, Vol. COM-28, 1980, pp. 711-719.

[10] Chazan, D., and Miranker, W., "Chaotic Relaxation," Linear Algebra and Applications, Vol. 2, 1969, pp. 199-222.

[11] Baudet, G.M., "Asynchronous Iterative Methods for Multiprocessors," Journal of the ACM, Vol. 2, 1978, pp. 226-244.

[12] Bertsekas, D.P., "Distributed Dynamic Programming," IEEE Trans. on Aut. Control, Vol. AC-27, 1982, pp. 610-616.

[13] Bertsekas, D.P., "Distributed Asynchronous Computation of Fixed Points," Math. Programming, Vol. 27, 1983, pp. 107-120.

[14] McQuillan, J., Falk, G., and Richer, I., "A Review of the Development and Performance of the ARPANET Routing Algorithm," IEEE Trans. on Communications, Vol. COM-26, 1968, pp. 1802-1811.

[15] Zangwill, W.I., Nonlinear Programming, Prentice Hall, Englewood Cliffs, N.Y., 1969.

[16] Luenberger, D.G., Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, MA, 1973.

[17] Ortega, J.M. and Rheinboldt, W.C., Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, N.Y., 1970.

[18] Polak, E., Computational Methods in Optimization: A Unified Approach, Academic Press, N.Y., 1971.

[19] Poljak, B.T., "Convergence and Convergence Rate of Iterative Stochastic Algorithms," Automation and Remote Control, Vol. 12, 1982, pp. 83-94.

[20] Bertsekas, D.P., Dynamic Programming and Stochastic Control, Academic Press, N.Y., 1976.

[21] Sandell, N.R., Jr., P. Varaiya, M. Athans and M. Safonov, "Survey of Decentralized Control Methods for Large Scale Systems," IEEE Trans. on Aut. Control, Vol. AC-23, No. 2, 1978, pp. 108-128.

[22] Ho, Y.C., "Team Decision Theory and Information Structure," IEEE Proceedings, Vol. 68, No. 6, 1980, pp. 644-654.

[23] Yao, A.C., "Some Complexity Questions Related to Distributed Computing," Proc. of the 11th STOC, 1979, pp. 209-213.

[24] Papadimitriou, C.H. and M. Sipser, "Communication Complexity," Proc. of the 14th STOC, 1982, pp. 196-200.

[25] Witsenhausen, H.S., "A Counterexample in Stochastic Optimum Control," SIAM J. Control, Vol. 6, No. 1, 1968, pp. 138-147.

[26] Papadimitriou, C.H. and J.N. Tsitsiklis, "On the Complexity of Designing Distributed Protocols," to appear in Information and Control, 1983.

[27] Tenney, R.R. and N.R. Sandell, Jr., "Detection with Distributed Sensors," IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-17, No. 4, 1981, pp. 501-509.

[28] Willsky, A.S., M. Bello, D.A. Castanon, B.C. Levy and G. Verghese, "Combining and Updating of Local Estimates and Regional Maps along Sets of One-Dimensional Tracks," IEEE Trans. on Aut. Control, Vol. AC-27, No. 4, 1982, pp. 799-813.

[29] Sanders, C.W., E.C. Tacker, T.D. Linton, R.Y.-S. Ling, "Specific Structures for Large Scale State Estimation Algorithms Having Information Exchange," IEEE Trans. on Aut. Control, Vol. AC-23, No. 2, 1978, pp. 255-261.

[30] Borkar, V. and P. Varaiya, "Asymptotic Agreement in Distributed Estimation," IEEE Trans. on Aut. Control, Vol. AC-27, No. 3, 1982, pp. 650-655.

[31] Arrow, K.J. and L. Hurwicz, "Decentralization and Computation in Resource Allocation," in Essays in Economics and Econometrics, R.W. Pfouts, Ed., Univ. of North Carolina Press, Chapel Hill, NC, 1960, pp. 34-104.

[32] Meerkov, S.M., "Mathematical Theory of Behavior-Individual and Collective Behavior of Retardable Elements," Mathematical Biosciences, Vol. 43, 1979, pp. 41-106.

[33] Astrom, K.J. and P. Eykhoff, "System Identification-A Survey," Automatica, Vol. 7, 1971, pp. 123-162.

[34] Solo, V., "The Convergence of AML," IEEE Trans. on Aut. Control, Vol. AC-24, 1979, pp. 958-963.

[35] Tsitsiklis, J.N., "Problems in Decentralized Decision Making and Computation," Ph.D. Thesis, Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, MA, in preparation.

[36] Nemirovsky, A.S., "On a Procedure for Stochastic Approximation in the Case of Dependent Noise," Engineering Cybernetics, 1981, pp. 1-13.

[37] Ljung, L., "Analysis of Recursive Stochastic Algorithms," IEEE Trans. on Auto. Control, Vol. AC-22, No. 4, 1977, pp. 551-575.

[38] Ljung, L., "On Positive Real Transfer Functions and the Convergence of Some Recursive Schemes," IEEE Trans. on Auto. Control, Vol. AC-22, No. 4, 1977, pp. 539-551.

[39] Tsitsiklis, J.N. and M. Athans, "Convergence and Asymptotic Agreement in Distributed Decision Problems," to appear in the IEEE Trans. on Aut. Control, 1984.

[40] Poljak, B.T. and Y.Z. Tsypkin, "Pseudogradient Adaptation and Training Algorithms," Automation and Remote Control, No. 3, 1973, pp. 45-68.

[41] Geanakoplos, J.D. and H.M. Polemarchakis, "We Can't Disagree Forever," Institute for Mathematical Studies in the Social Sciences, Technical Report No. 277, Stanford University, Stanford, CA, 1978.


FILM